Software Development Recommendations for IOC-1

K. Berkbigler, B. Bush, J. Davis

Los Alamos National Laboratory

30 January 1995

Outline

- Development environment
- Tools
- Major risks
- Cost estimate
- Software engineering practices

DEVELOPMENT ENVIRONMENT

Supported Platforms

- The initial development of TRANSIMS will take place on the Sun hardware/software platform.
- The development work will be portable to other Unix platforms, but not necessary immediately, insofar as compiler technology currently varies widely from platform to platform.
- Windows NT will be supported for the user interface, but not for the planner and microsimulation.
- Windows, DOS, and MacIntosh will not be directly supported, although they may be used as X clients for the user interface.

Software

- The SunPro C++ suite of C++ development tools will be used for work on the Sun hardware/software platform. This suite includes a compiler, debugger, source code editor, and object-oriented browser.
- The Borland integrated C++ development environment will be used for Windows NT development.
- Both the SunPro and Borland compilers are certified for use with the C++ class libraries recommended below.
- The configuration management tool's version of "make" will be used for building executables.
- Purify will be used for debugging memory management.
- The word processor standard will be FrameMaker.

- The standard tool for object-oriented software engineering will be Rose/C++.
- The configuration management tool will be ClearCase.

Hardware

- An additional 10 GB of disk space will be required at Pueblo School for IOC-1.
- Tape drive support of 30 GB will also be necessary.
- Note that these are estimates based on immediately foreseeable needs.

TOOLS (THIRD-PARTY SOFTWARE)

C++ Libraries

- The ANSI standard C library and the ANSI draft standard C++ library will be used. POSIX-compliant operating system calls will also be used. This combination of standard libraries will support the portability between Unix platforms, and between Unix and Windows NT.

- The Booch Components will be used for container classes and exception handling. These are robust, easily usable, well-designed, and portable.

- PVM will be used for distributed application development; it is portable between Unix platforms.

- RPC will be used to support interprocess communication; it is portable between Unix platforms, and between Unix and Windows NT.

- The Rogue Wave DBtools.h++ library will be used for database access. It provides a C++ interface to databases such as Oracle and Sybase.

Graphical User Interface

- ArcView will be the primary graphical user interface; it will be customized with Avenue to provide a uniform interface to as many of the TRANSIMS subsystems as possible. It will serve as a launcher, viewer, and data manipulator.

- It may be necessary to use ArcInfo for the parts of the graphical user interface where the customization capabilities of Avenue are insufficient.

- No specialized software will be purchased at this time to support animation. Thus a GUI builder will not be needed presently.

- Specialized plotting software for making the types of plots that ArcView and S+ do not support will not be purchased at this time.

Database

- We will use the Oracle7 relational database to manage the majority of the data in the database subsystem. This interfaces seamlessly with ArcView and ArcInfo.

- A customized storage system will be required to compress large simulation output data sets to manageable size.

Statistics

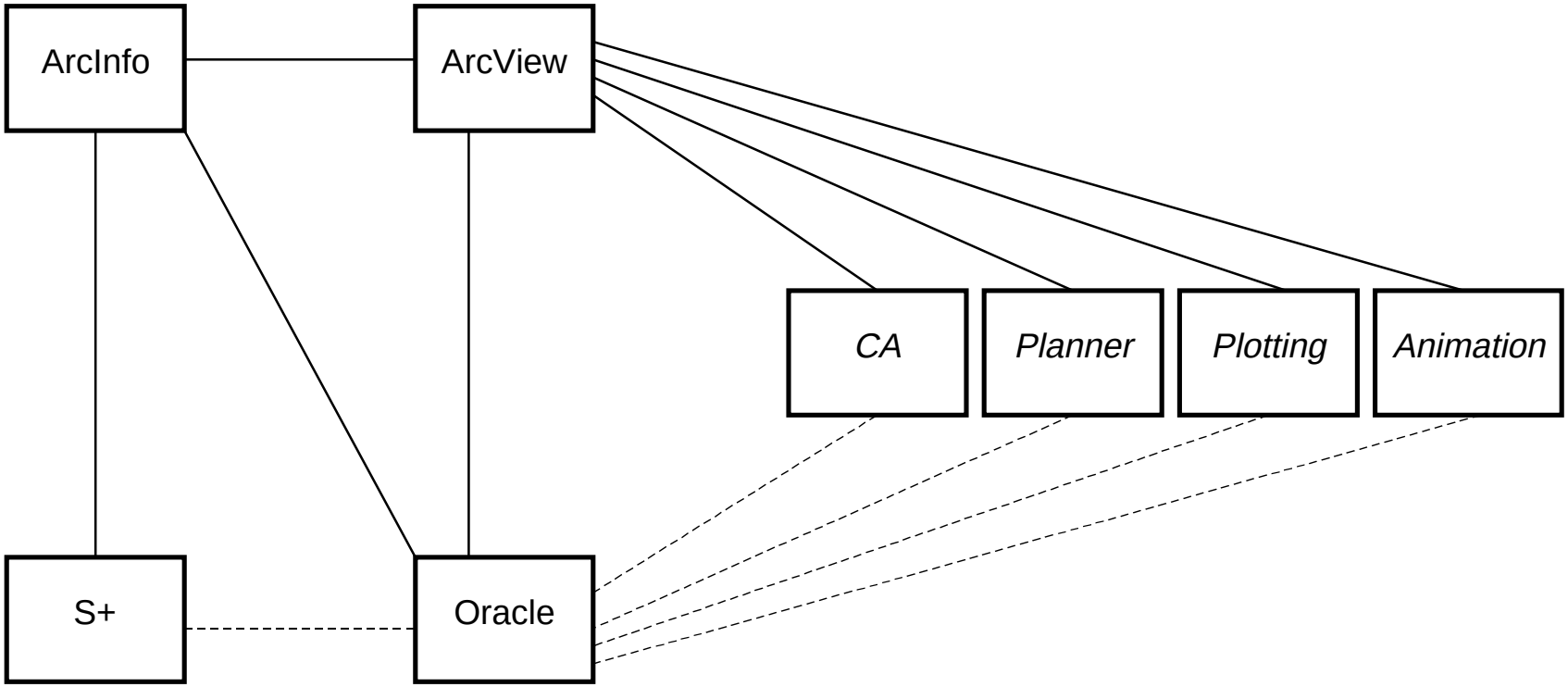
- The ArcInfo-compatible version of S+ will be used for statistical analysis.

Geographic Information System

- The Unix version of ArcInfo 7 will be used for the processing and analysis of geographic data; one floating license will be required. We will also purchase the network, and grid packages.

- The majority of the GIS functionality will be in ArcView, with ArcInfo used as a calculation engine.

Software Executables Overview



In the above diagram, the solid lines represent strong links and the dashed lines represent weak links. The names in regular type are commercial products that will be customized; the names in italic type are executables that will be written in C++ by the software development team.

MAJOR RISKS

Portability

- The TRANSIMS software may not be immediately portable from the Sun and Windows NT platforms to other Unix or POSIX platforms.
- The necessary software support (compilers, libraries, tools) may not be currently available for some of these platforms.
- However, we believe that TRANSIMS will be highly portable in the long run.

Limited Product Research

- The software team has not formally evaluated many of the software products that we recommend using for TRANSIMS.
- Thus, the products may have unforeseen limitations or performance problems that will adversely affect the software development process.
- One important example of this is our reliance on ArcView (and to a lesser extent, on ArcInfo) to provide the graphical user interface for TRANSIMS—we have performed no prototyping in this area to verify that all of the required capabilities are present.

Integration

- Although we have verified that ArcView, ArcInfo, Oracle, and S+ can work together as an integrated set of tools, we do not know the difficulty of achieving seamless integration.

- Also, we have not explored all of the issues relating to the integration of commercial products with the C++ code that we write for the planner, simulator, etc.

Data Storage

- We have not performed any experiments to determine how products such as Oracle will perform for TRANSIMS applications.

- All of the data storage options that we considered (relational database, object-oriented database, b-trees, ArcInfo files, binary files, text files) have drawbacks in terms of either access speed, file size, ease of use, cost, development time, or connectivity.

COST ESTIMATE

Software

Product	*Licenses*	*Type*	*Cost*
ArcView+Avenue	3	floating	$9k
ArcInfo	1	floating	$7k
ArcInfo grid	1	floating	$1k
ArcInfo network	1	floating	$1k
S+	1	floating	$4k
Oracle	8	concurrent	$10k
Dbtools.h++	3	developer	$3k
Booch Components	6	developer	$3k
SunPro C++	5	site	$1k
Rose	1	floating	$5k
ClearCase	6	user	$14k
Total			*$58k*

Hardware

Item	*Cost*
10 GB hard disk	$10k
30 GB tape storage	$8k
Total	*$18k*

Training

- We recommend that up to $10k be available for the attendence of training classes by software team members.

SOFTWARE ENGINEERING PRACTICES

Standards: *Coding*

- We recommend adopting the coding (*naming* and *usage*) standards presented in *Taligent's Guide to Designing Programs*.

- Supplemental usage guidelines are available in the books by Cline & Lomow (*C++ FAQs*), Murray (*C++ Strategies and Tactics*), Cargill (*C++ Programming Style)*), Meyers (*Effective C++*), Coplien (*Advanced C++*), and Stroustrup (*The C++ Programming Language*).

- Additional coding recommendations are:
  - Classes have names indicating in which subsystem they belong. This will aid navigating source code.
  - Formal parameters for member functions should be included in the class definitions. This makes the class definition more understandable.
  - Member functions should not be inlined in the class definition. This makes the definition less cluttered and makes it easily to switch from inlining to not inlining.

- Recommendations for source documentation are:
  - The documentation for each class should include a general description of the services it provides and specify any invariants or states it has.
  - The documentation for each member function should include a general description of what it does, define its arguments, specify its pre- and post-conditions, say what state changes may occur, and specify what exceptions it throws and why.

Standards: *Documentation*

- The TRANSIMS documentation will describe TRANSIMS at a level of detail that is useful to the software developer and at the same time understandable by the non-programmer. It will consist primarily of English text, supported and augmented by the diagrams discussed below. Refer to Iseult White, *Using the Booch Method*, for more detail.

- The design documentation will include the following:
  - The *architectural specification* captures the major abstractions and design decisions that apply to the whole system.
  - *Class-category diagrams* partition the system into high-level groupings of classes and show their visibility to each other.
  - *Design class diagrams* identify the key classes of the domain and show the relationships between classes. They also show the abstractions of the physical implementation, detailed data types and structures, and the mapping of the logical abstractions to the physical abstractions.
  - *Design class specifications* define the class, its relationships, its attributes, its superclasses, and the interface of its operations. They also show

domain and implementation details, such as algorithms for complex operations, internal data attributes, and access control for operations and data.
  - *Design object-scenario diagrams* illustrate how the objects will interact when tracing the execution of a use case. They also show the full implementation detail of a key mechanism, including objects that deal with I/O, data structures, and persistent data.

- The implementation documentation will consist of the following:
  - *Documented source code.* Source code is documented according to standards defined in our previous discussion.
  - *Test results.* Results from carrying out the test suite are saved for regression testing in subsequent executable releases.
  - *User manuals.* User documentation provides sufficient detail for the non-programmer to successfully exercise the software. The documentation will be available on-line.

Standards: *Configuration Management*

- All source code, data files, diagrams, and documents should be under configuration management control where practical.
- One team member should be the primary contact for dealing with configuration management issues.
- A configuration management plan should be completed as soon as possible.

Process: *Methodology*

- We recommend the Booch methodology as presented in John Berry's class notes, Iseult White's primer, and Grady Booch's text.

- To reduce risk, we emphasize the following points from "Managing Development," chapter 8 of John Berry's class notes:
  - We must fully utilize iterative development; we should have an incrementally improved executable release every three to six weeks.
  - We must preserve a clean, layered internal system structure (architecture).
  - The individuals responsible for the TRANSIMS architecture must continue to maintain the architectural integrity of the system. Specifically, the team should approve all changes to architectural interfaces, help assess project risks, and help schedule the order and the content of the iterations.

- To coordinate the development effort, we will use Iseult White's book as a primer, especially for identifying project deliverables. Although we will produce simplified versions of some of the documents, we should follow all the

steps of each iterative cycle and produce a new executable version every few weeks.

- For efficiency we should use the Rose tool as fully as possible and stick to its version of Booch notation and C++ coding style. These match White's book. We should rely on the professionalism of the team members rather than spend time overspecifying the methodology.

- Design reviews will help us decide how far we want to go with things like Booch "adornments."

Process: *Reviews*

- We recommend the adoption of a software inspection procedure along the lines of that outlined by Steve McConnell in §24.2 of *Code Complete.*
  - Each inspection involves a moderator/scribe, author, and 1-4 reviewers.
  - Each reviewer works alone inspecting the code for 90 minutes.
  - Later the participants meet and the author paraphrases the code design and logic.
  - Errors and their severity are noted by the reviewers, but they are not discussed beyond identifying them.
  - The length of the meeting is limited to two hours and the moderator writes an inspection report.
  - Informal discussion of solutions is postponed until after the meeting.

- Reviews will be held for both the design and the implementation phases of development.

- Reviewers external to TRANSIMS may be included on the review panels.

Process: *Testing*

- It is recommended that each unit (class or group of classes) be tested by the author or other interested parties before the code is checked into the configuration management system for use by others.
- All test documentation, code, and data will be under configuration management control.
- All functions and branches within functions will be tested.
- Integration testing will take place as soon as sufficient units are available to integrate the components.
- System testing will take place once per iteration of the software development cycle.